VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozšíření programu Neural Racer

Extension of Neural Racer Program

2012

Petr Hamalčík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Petr Hamalčík**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Rozšíření programu Neural Racer
Extenzion of Neural Racer Program

Zásady pro vypracování:

Cílem práce je rozšířit stávající simulační program pro automobilové závody neuronových sítí v řízení auta po stanovené dráze. K simulačnímu programu se připojují jednotlivé neuronové sítě prostřednictvím internetu k závodnímu serveru a snaží se projet zadanou trať v co možná nejkratším čase.

Program bude rozšířen o:
1. Nový reálnější fyzikální model.
2. Možnost umisťovat na trať překážky a bariéry.
3. Zjišťování překážek před autem.
4. Předjíždění a kolize mezi auty.
5. Navigace auta ke kontrolním bodům.
6. Tratě s různými typy povrchů.
7. Auta s rozdílnými jízdními vlastnostmi.
8. Autentizace studentů přes LDAP, šampionáty, bodování a žebříčky nejlepších jezdců.
9. Rozšíření komunikačního protokolu o přenos nových informací o trati.

Seznam doporučené odborné literatury:

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides (Gang of Four): Návrh programů pomocí vzorů. Grada. Praha 2003. ISBN 8024703025
VONDRÁK, Ivo. Neuronové sítě. Ostrava : VŠB - TU Ostrava, 1995. 56 s. Dostupné z WWW: <http://vondrak.cs.vsb.cz/download/Neuronove_site.pdf>.
Dále podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 18.11.2011

Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 17. srpna 2012 ..... Hanzlík .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 17. srpna 2012 ..... Hanzlík .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, zejména pak vedoucímu Ing. Davidu Ježkovi, za cenné rady a čas, který mi během tvorby této práce věnoval.

Abstrakt

Tato práce popisuje rozšíření programu pro simulaci závodů aut řízených neuronovými sítěmi. Neuronové sítě implementují studenti v rámci výuky. Rozšíření programu zahrnuje integraci fyzikální knihovny JBullet, přidání nových funkcí, zdokonalení uživatelského rozhranní a další vylepšení.

Klíčová slova: závody aut, umělá inteligence, neuronová síť, fyzikální knihovna JBullet

Abstract

This thesis describes extension of a program for car racing simulation. Cars are controlled by neural networks implemented by students within scope of studies. System extension includes intergration JBullet physics library, adding new features, user interface improvement and other upgrades.

Keywords: car racing, artificial intelligence, neural network, JBullet physics library

Seznam použitých zkratek a symbolů

AABB	–	Axis Aligned Bounding Box
API	–	Application Programming Interface
GUI	–	Graphical User Interface
JAR	–	Java ARchive
JRE	–	Java Runtime Environment
LDAP	–	Lightweight Directory Access Protocol
LWJGL	–	Lightweight Java Game Library
SSL	–	Secure Sockets Layer
TCP	–	Transmission Control Protocol
XML	–	Extensible Markup Language

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Cílem této práce je rozšířit stávající systém pro simulaci závodů aut řízených neuronovými sítěmi. Tento program funguje na bázi klient-server. Na serveru běží simulátor závodů, ke kterému se připojují klienti ovládající jednotlivá vozidla. Klienty implementují studenti v rámci předmětu neuronové sítě. Úkolem virtuálních závodníků je sledovat středovou čáru a projet trať v co nejkratším čase.

Součástí práce je stručný popis fungování neuronových sítí a představení původního programu. Dále jsou popsána vylepšení aplikace. Do simulace závodů jsem integroval fyzikální knihovnu JBullet, která simuluje jízdní model automobilů a zajišťuje detekci kolizí mezi vozidly a překážkami. Díky tomu je možné vytvářet tratě s různými typy povrchů a umísťovat na trať objekty, kterým se auta snaží vyhýbat. Vylepšením prošel systém organizace závodů na serveru, editor tratí a testovací režim. Studenti se mohou přihlašovat přes LDAP a po přihlášení se ukládají jejich výsledky do databáze. Vylepšena byla také 2D grafika prohlížeče závodů a uživatelské rozhraní.

2 Neuronové sítě

Umělá neuronová síť[1] je matematický model, který se snaží napodobit chování biologických neuronů nacházejících se v mozku. Neuronová síť je struktura složená s navzájem propojených neuronů. Znalosti jsou ukládány pomocí síly vazeb mezi jednotlivými neurony. Neuronové sítě jsou určené pro distribuované paralelní zpracování dat a jejich podstatnou vlastností je schopnost učit se.

2.1 Spojitý perceptron

Základním stavebním kamenem neuronových sítí je model neuronu zvaný perceptron. Perceptron má libovolné množství vstupů x a jeden výstup y. Každý vstup má váhu w. Hodnota výstupu je dána excitační funkcí $S(z)$. Tato funkce je závislá na vnitřním potenciálu z. Vnitřní potenciál je dán váženým součtem vstupů, od kterého je odečten práh neuronu θ. Pro zjednodušení je práh neuronu převeden na nultý vstup $x_0 = 1$ s váhou $w_0 = -\theta$.

Pro hodnotu z platí vztah:

$$z = \sum_{i=0}^{n} w_i x_i$$

kde jsou:

- $x_1 ... x_n$ ... hodnoty vstupů
- $w_1 ... w_n$ ... hodnoty vah
- $x_0 = 1$ ... nultý vstup
- $w_0 = -\theta$ ... nultá váha

Hodnota samotné excitační funkce je dána tímto vztahem:

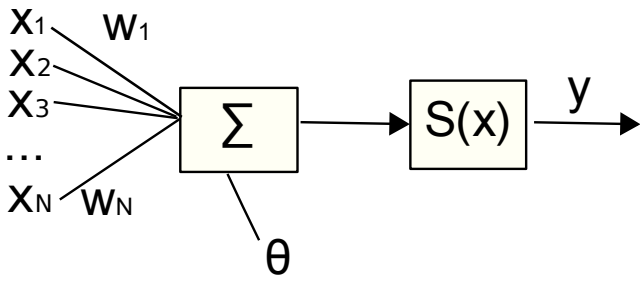
$$y = S(z) = \frac{1}{1 + e^{-\lambda z}}$$

kde λ značí strmost sigmoidu.

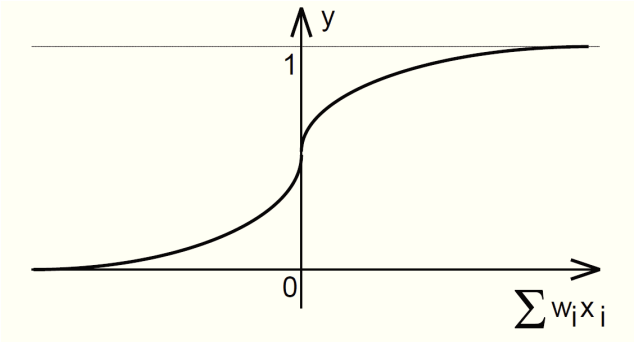
Učení perceptronu je proces, kdy dochází k nastavení vah tak, aby dokázal rozpoznávat předložené vstupy. Nejznámějším algoritmem učení je tzv. Hebbovo pravidlo:

1. Inicializace vah a prahu náhodnými malými čísly
2. Předložení požadovaného vstupu
3. Spočítání reálné hodnoty výstupní funkce
4. Adaptace vah v závislosti na reálném a požadovaném výstupu neuronu

Excitace neuronu se pohybuje v rozmezí 0 až 1. Nulová hodnota znamená úpné utlumení, 1 znamená maximální excitaci neuronu.



Obrázek 1: Neuron [2]



Obrázek 2: Excitační funkce neuronu [1]

2.2 Vícevrstvá neuronová síť a metoda backpropagation

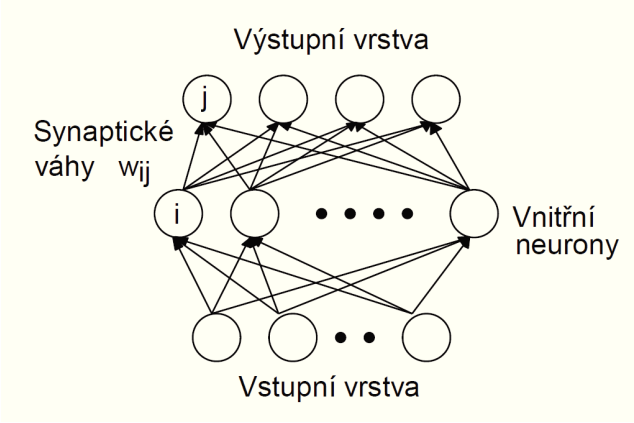
Vzájemně propojené spojité perceptrony tvoří neuronovou síť. Nejrozšířenějším modelem je vícevrstvá síť, která se skládá ze vstupní vrstvy, výstupní vrstvy a alespoň jedné vnitřní vrstvy. Mezi sousedními vrstvami se nachází úplné propojení neuronů.

Odezvu neuronové sítě získáme dopředným šířením, které probíhá tímto způsobem:

1. Nejprve jsou neurony vstupní vrstvy excitovány hodnotami v rozmezí 0 až 1.
2. Tyto excitace jsou pomocí vazeb přivedeny k následující vrstvě a upraveny (zesíleny či zeslabeny) pomocí synaptických vah.
3. Každý neuron této vyšší vrstvy provede sumaci upravených signálů od neuronů nižší vrstvy a je excitován na úroveň danou svou aktivační funkcí.
4. Tento proces probíhá přes všechny vnitřní vrstvy až k vrstvě výstupní, kde pak získáme excitační stavy všech jejích neuronů.

K učení neuronové sítě je potřebná tzv. trénovací množina. Ta se skládá z jednotlivých učebních vzorů. Vzor je definován jako dvojice: I_i, O_i, kde I_i je vektor excitací vstupní vrstvy a O_i je vektor očekávaných excitací výstupní vrstvy.

Metoda, která umožňuje adaptaci neuronové sítě nad danou trénovací množinou, se nazývá backpropagation. Spočívá v šíření signálu od výstupní vrstvy zpět k vrstvám nižším. Proces učení probíhá takto:



Obrázek 3: Vícevrstvá neuronová síť [1]

1. Vezmeme nejprve vektor I_i i-tého prvku trénovací množiny, kterým excitujeme neurony vstupní vrstvy na odpovídající úroveň.

2. Známým způsobem provedeme dopředné šíření tohoto signálu až k výstupní vrstvě neuronů.

3. Rozdíl mezi skutečnou a požadovanou odezvou definuje chybu neuronové sítě. Tuto chybu pak v určitém poměru - learning rate - „vracíme zpět“ do neuronové sítě formou úpravy synaptických vah mezi jednotlivými vrstvami směrem od horních vrstev k vrstvám nižším tak, aby chyba při následující odezvě byla menší.

4. Tento proces probíhá přes všechny vnitřní vrstvy až k vrstvě výstupní, kde pak získáme excitační stavy všech jejích neuronů.

5. Po vyčerpání celé trénovací množiny se vyhodnotí celková chyba přes všechny vzory trénovaci množiny. Pokud je chyba vyšší než požadovaná, celý proces se opakuje znovu.

Po naučení by síť měla mít schopnost generalizace - tedy měla by umět předvídat hodnotu výstupní funkce pro vstupy, které nebyly součástí učení, ale lze si je nějakým způsobem odvodit ze vzorů trénovací množiny.

3 Původní program

Původní program [2] pro simulaci závodů neuronových sítí je vytvořen v jazyce Java. Skládá se ze dvou částí: serveru a klienta-prohlížeče. Obě části jsou v jednom JARu. Server je možné spustit parametrem příkazové řádky nebo z grafického rozhraní prohlížeče. K serveru se může připojit klient-řidič, tedy studentem vytvořený program s neuronovou sítí řídící auto. V prohlížeči je možné organizovat a sledovat závody. Součástí aplikace je i editor tratí. Server komunikuje s klienty pomocí textových zpráv na protokolu TCP.

Úkolem auta je sledovat středovou čáru a projet v co nejkratším čase všechny kontrolní body, které jsou umístěné na trati. Auta startují ze stejného místa a navzájem nekolidují. Jízda mimo trať auto zpomaluje a při prudkém zatáčení při vysoké rychlosti může dostat smyk.

Klient-řidič dostává v průběhu závodu informace o poloze auta převedené na vstupy neuronové sítě:

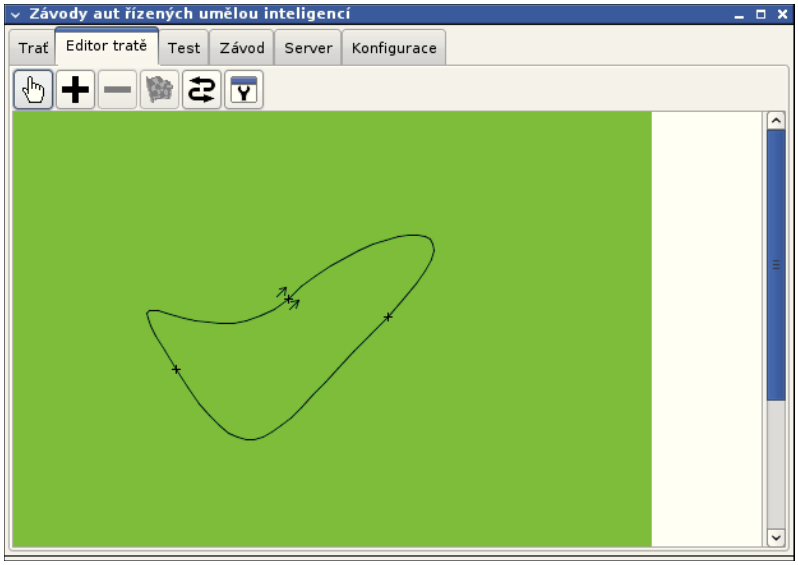
- distance – vzdálenost auta od středové čáry
- angle – úhel mezi čárou a směrem jízdy
- speed – relativní rychlost auta
- distance2 – vzdálenost od čáry za určitou časovou jednotku, pokud bude auto pokračovat v jízdě stejným směrem

Po odeslání informací o poloze auta očekává server odpověď od řidiče:

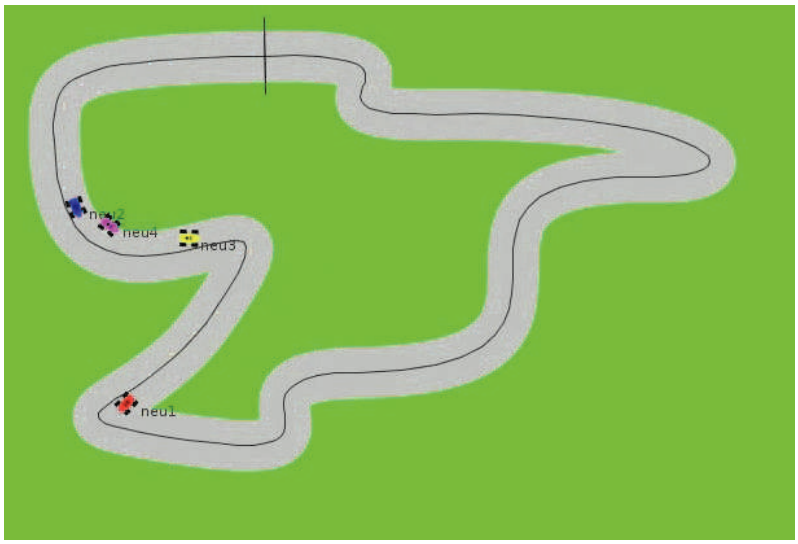
- wheel – natočení volantu
- acc – zrychlení auta

GUI programu běží v jednom okně a jednotlivé části jsou rozdělené do panelů podle funkce:

- Trať – Vytváření, načítání a ukládání tratě lokálně nebo na serveru.
- Editor tratě – Editor, kde je možné upravovat trať reprezentovanou Beziérovou křivkou (obr. 4).
- Test – Testovací režim, který umožňuje připojení řidiče a jeho testování na trati otevřené v editoru.
- Závod – Umožňuje vytvoření nebo připojení se k závodu na serveru. Po skončení závodu je nutné odpojit se. Probíhající závod je vidět na obr. 5
- Server – Spuštění nebo zastavení závodního serveru.
- Nastavení – Nastavení programu



Obrázek 4: Editor tratě [2]



Obrázek 5: Probíhající závod [2]

4 Fyzikální model

4.1 Fyzikální knihovna JBullet

Jako první byl vylepšen fyzikální model. Původní program neobsahuje téměř žádnou fyziku - auto jede tam, kam má otočena kola, a pokud zrychlení, úhel zatáčení a rychlost překročí určitou hodnotu, dostane se do smyku. Rozhodl jsem se, že než se pokoušet vytvořit fyzikální model sám, bude lepší integrovat do programu některou open source fyzikální knihovnu. Pro programovací jazyk Java jsou v současné době dostupné například tyto knihovny:

- JBox2D
- Phys2D
- JBullet
- Ode4j

JBox2D a Phys2D simulují, jak už název napovídá, fyziku ve dvou rozměrech. Obě jsou portem C++ knihovny Box2D do jazyku Java. Náš program je sice pouze 2D, ale pro jeho možnou rozšiřitelnost jsem se rozhodl, že bude leší použít trojrozměrnou fyziku. Navíc JBox2D ani Phys2D nemají přímou podporu pro vozidla. Dále jsem se rozhodoval mezi 3D knihovnami JBullet [3] a Ode4j. Nakonec zvítězil JBullet, hlavně díky propracovanější simulaci vozidel.

JBullet je Java portem open source knihovny Bullet 2.72 (aktuální C++ verze je 2.80) a obsahuje většinu hlavních funkcí C++ verze. JBullet nemá vlastní manuál, ale je možné použít ten z původního Bulletu [4]. Je však třeba počítat s tím, že některé pokročilé funkce nejsou ve verzi pro Javu implementované. Pro použití JBulletu v projektu je třeba importovat knihovny *jbullet.jar* a *vecmath.jar*.

4.2 Základy simulace v JBullet

Základem fyzikální knihovny je simulace dynamiky pevných těles a detekce kolizí mezi nimi. Aby bylo možné provádět simulaci, je nutné nejdřív vytvořit fyzikální „svět“, tedy instanci třídy `DiscreteDynamicsWorld`. Té se předávájí parametry, které určují jak se bude řešit detekce kolizí a interakce mezi tělesy:

- `BroadphaseInterface` - určuje jaký algoritmus se použije v první fázi detekce kolizí, kdy se mají rychle zjistit potenciálně kolidující páry objektů. K testování kolizí se používají kvádry obalující tělesa (AABB). V následující fázi označované jako narrowphase se testuje, jesli objekty skutečně kolidují.
- `CollisionConfiguration` - obsahuje algoritmy k testování skutečných kolizí mezi objekty.
- `Dispatcher` - pro každý pár potenciálně kolidujících objektů vybere vhodný algoritmus podle jejich tvarů a zjistí jestli skutečně nekolidují.

- `ConstraintSolver` - řeší interakce mezi tělesy, bere v potaz gravitaci, působení sil, kolize a vzájemné propojení těles.

```
BroadphaseInterface overlappingPairCache = new DbvtBroadphase();
CollisionConfiguration collisionConfiguration = new DefaultCollisionConfiguration();
Dispatcher dispatcher = new CollisionDispatcher(collisionConfiguration);
ConstraintSolver constraintSolver = new SequentialImpulseConstraintSolver();
DynamicsWorld dynamicsWorld = new DiscreteDynamicsWorld(dispatcher, overlappingPairCache,
    constraintSolver, collisionConfiguration);
```

Výpis 1: Vytvoření fyzikálního světa

Do vytvořeného světa je možné přidat pevná tělesa. Pevné těleso má určitý tvar, hmotnost a pozici v prostoru. JBullet rozlišuje tyto typy těles:

Dynamické pevné těleso - má kladnou hmotnost, může se pohybovat, působí na něj síly.

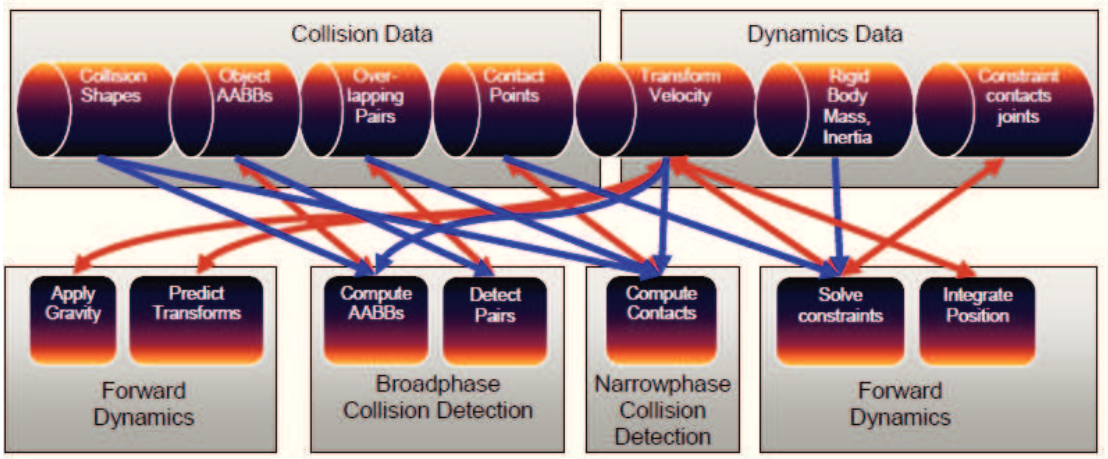
Statické pevné těleso - má nulovou hmotnost, nemůže se pohybovat.

Kinematické pevné těleso - má nulovou hmotnost, pohybuje s ním uživatel, nepůsobí na něj síly.

Pevné těleso by mělo mít přiřazeno určitý tvar, který slouží k detekci kolizí, a v případě dynamického tělesa určuje rozložení hmotnosti. Všechny třídy reprezentující tvary dědí ze třídy `CollisionShape`. Základní tvary je navíc možné kombinovat do složeného tvaru `CompoundShape`. JBullet umožňuje vytvářet tělesa s těmito tvary:

- `SphereShape` – koule
- `BoxShape` – kvádr určený třemi rozměry
- `CylinderShape` – válec
- `CapsuleShape` – válec zakončený polokoulemi
- `ConeShape` – kužel
- `ConvexHullShape` – Konvexní tvar definovaný množinou bodů v prostoru. Nejmenší plášť, který tyto body obaluje, tvoří povrch tělesa.
- `BvhTriangleMeshShape` – Statická trojúhelníková síť vhodná k vytvoření scenérie, může obsahovat velké množství trojúhelníků.
- `StaticPlaneShape` – statická nekonečná rovina

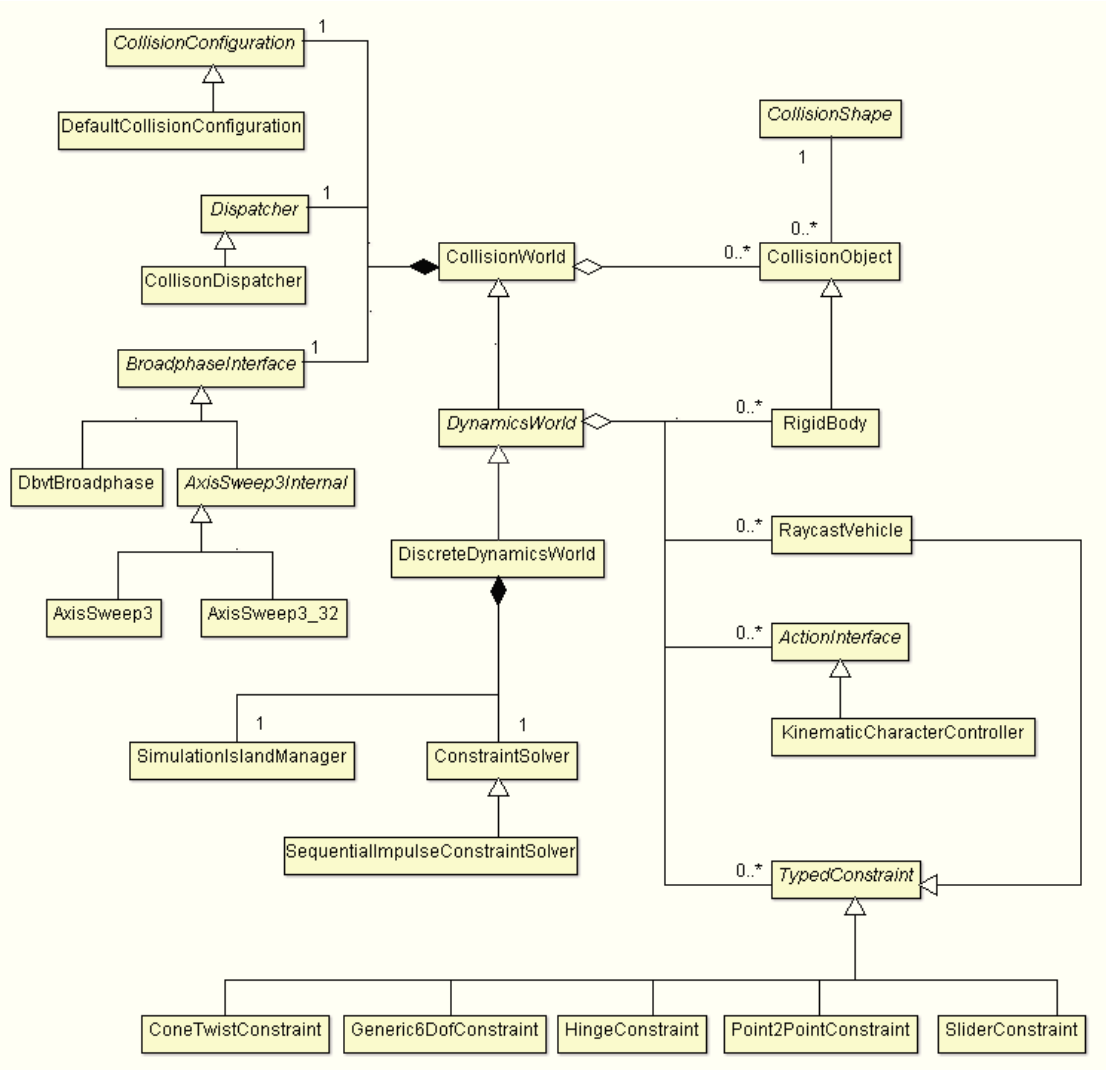
Krok simulace se provede voláním metody `stepSimulation` s parametry `timeStep`, `maxSubSteps` a `fixedTimeStep`. První parametr je povinný a udává o kolik sekund se čas posune v simulaci. Další dva jsou nepovinné. Druhý určuje kolik mezikroků simulace se provede maximálně při jednom volání metody. Poslední parametr je určuje velikost vnitřního kroku v JBulletu a má základní hodnotu $\frac{1}{60}$ sekundy.



Obrázek 6: Krok simulace [4]

Pokud je parametr `timeStep` větší než `fixedTimeStep`, musí se zvýšit hodnota `maxSubSteps`. Např. pokud je krok simulace $\frac{1}{10}$ sekundy, musí JBullet provést aspoň 6 mezikroků při každém volání metody (při základní velikosti vnitřního kroku). V programu Neural Racer je možné nastavit je možné nastavit počet kroků za sekundu od 10 do 60. Co se provádí v každém kroku simulace je vidět na obrázku 6.

Na obr. 7 je vidět diagram s hlavními třídami v JBulletu. `CollisionWorld` zjišťuje kolize objektů reprezentovanými třídou `CollisionObject`. `CollisionShape` definuje rozměry a tvar objektu. Abstraktní třída `DynamicsWorld` rozšiřuje `CollisionWorld` o dynamiku pevných těles. Konkrétní implementace je `DiscreteDynamicsWorld`. Do dynamického světa je možné přidat pevná tělesa `RigidBody`. `TypedConstraint` reprezentuje pohyblivé propojení mezi jednotlivými tělesy (např. klouby, panty u dveří). Speciálním typem `TypedConstraint` je `RaycastVehicle` simulující vozidlo. Abstraktní třída `ActionInterface` slouží k pohybování s kinematickými tělesy. `KinematicCharacterController` je implementace této třídy, která umožňuje ovládání postav.



Obrázek 7: Architektura jádra knihovny JBullet

4.3 Simulace vozidla

Bullet používá pro simulaci vozidel metodu vysílání paprsků, tzv. raycasting. Ta spočívá v tom, že pneumatiky a tlumiče auta nejsou simulovány jako tělesa, pouze se vyšle paprsek z místa připojení tlumiče k šasi směrem k vozovce. Podle matematického modelu se poté vypočte stlačení tlumiče a pozice karoserie nad zemí. Vozidlo reprezentuje instance třídy `RaycastVehicle`.

Samotná karoserie vozidla je reprezentována jedním pevným tělesem, jehož tvar může být složený z více entit. Toto těleso se používá na detekci kolizí auta s ostatními objekty. V programu Neural Racer jsem použil pro reprezentaci vozidla základní kvádr, který je dán svými rozměry:

- `carWidth` – šířka auta
- `carHeight` – výška karoserie (bez kol)
- `carLength` – délka auta

Po vytvoření karoserie je potřeba nastavit tlumiče. K tomu slouží objekt `VehicleTuning` s těmito parametry:

- `maxSuspensionTravelCm` – Maximální stlačení tlumičů v centimetrech.
- `suspensionStiffness` – Tuhost tlumičů. Orientační hodnoty jsou 10 pro terénní buginu, 50 pro sportovní auto, 200 pro formuli 1.
- `suspensionDamping` – Tlumivost odpružení při roztahování. Vypočte se podle vztahu $2k_d\sqrt{s}$, kde s je tuhost tlumičů a k_d je koeficient nabývající hodnot 0 až 1, doporučené hodnoty jsou 0,2 až 0,5.
- `suspensionCompresion` – Tlumivost odpružení při stlačování. Vypočte se podle vztahu $2k_c\sqrt{s}$, kde s je tuhost tlumičů a k_c je koeficient nabývající hodnot 0 až 1, doporučené hodnoty jsou 0,1 až 0,3. Hodnota k_c by měla být trochu menší než k_d.
- `frictionSlip` – Určuje součinitel tření pneumatik, který by se měl nastavit podle povrchu vozovky. V sekci 4.5 to je popsáno podrobněji.

Umístění a rozměry tlumičů určují v mém modelu následující parametry:

- `connectionHeight` – Výška připojení tlumičů k šasi.
- `wheelBase` – Rozchod (vzdálenost mezi nápravami).
- `wheelTrack` – Rozvor (vzdálenost mezi pravým a levým kolem).
- `suspensionRestLength` – Délka nezatížených tlumičů.

Nakonec umístíme na tlumiče pneumatiky definované rozměry:

- `wheelRadius` - Poloměr kola.

- `wheelWith` - Šířka kola, neovliňuje chování auta, pouze grafickou reprezentaci.

Při simulaci vozidel jsem narazil na jeden problém - maximální síla, která může působit na jeden tlumič, je napevno nastavená na 6000N. Pokud je vytvořeno vozidlo, jehož váha převyšuje 2,4 tuny, jeho kola se propadají do země. Proto jsem do třídy RayCastVehicle z původního JBulletu přidal další nastavitelný parametr `gMaxSuspensionForce`, který lze použít k nastavení maximální síly u těžších vozidel. Při nastavování vozidla jsem vycházel z dokumentu [5].

4.4 Pohon vozidla

Pokud chceme, aby se vozidlo rozjelo, musí kola působit silou na vozovku. Tuto sílu produkuje motor vozidla a následně ji přenáší přes hnací ústrojí ve formě točivého momentu. Točivý moment způsobuje otáčení hřídele a vzniklá síla uděluje vozidlu zrychlení. Sílu, kterou působí kola na vozovku vypočteme, podle vzorce:

$$F = \frac{tTgde}{r}$$

kde značí:

- t - sešplápnutí plynového pedálu od 0 do 1
- T - točivý moment motoru [Nm]
- g - převodový poměr
- d - poměr diferenciálu
- e - účinnost hnacího ústrojí [%]
- r - poloměr pneumatiky [m]

V reálném autě závisí točivý moment na otáčkách motoru a určuje jej křivka točivého momentu, jejíž průběh je pro každý motor jiný. Ve svém modelu jsem tento problém zjednodušil - motor vždy produkuje maximální točivý moment, který jediný určuje výkon motoru.

Převodový poměr je závislý na zařazeném rychlostním stupni. Nižší stupně mají vyšší převodový poměr a přenášejí tak větší sílu, která vytváří větší zrychlení. Vyšší stupně přenášejí více otáček, což umožňuje autu dosáhnout větší rychlosti. V mém modelu funguje řazení jako automatická převodovka. Při dosažení určitých otáček motoru se zařadí vyšší stupeň, pokud naopak otáčky klesnou pod danou hranici, přeřadí se na nižší stupeň. Z toho je zřejmé, že pro fungování převodovky je třeba znát otáčky motoru. Ty lze vypočítat ze vztahu:

$$rpm = \frac{60\omega gd}{2\pi}$$

kde značí:

- ω - úhlová rychlost kola [rad/s]
- g - převodový poměr
- d - poměr diferenciálu

Úhlovou rychlost kola je možné nejjednodušeji spočítat z rychlosti auta a poloměru kola:

$$\omega = vr$$

Pokud na vozidlo působí síla motoru, začne zrychlovat. V reálném světě je hlavním limitujícím faktorem zrychlení odpor vzduchu. Ten působí proti směru pohybu vozidla. Protože odpor vzduchu roste se čtvercem rychlosti, síla motoru a odporová síla se po nějaké době vyrovnají a vozidlo přestane zrychlovat - dosáhne maximální rychlosti. Odpor vzduchu lze vypočítat ze vztahu:

$$F = \frac{1}{2}C_d\rho v^2 S)$$

- C_d - součinitel odporu vzduchu
- ρ - hustota vzduchu [kg/m^3]
- v - rychlost auta [m/s]
- S - plocha, na kterou síla působí, v našem případě výška x šířka karoserie

Do fyzikálního modelu se síla motoru přenáší jako parametr metody `applyEngineForce` třídy `RaycastVehicle`. Celková síla motoru se vydělí počtem poháněných kol a poté aplikuje na každé kolo s pohonem. Maximální brzdná síla má konstantní velikost a sílu brzd určuje jen stlačení brzdového pedálu. Brzdy se aktivují metodou `setBrake`. Síla odporu vzduchu působí proti směru pohybu auta a předává se jako parametr metody `applyCentralForce`. V každém kroku simulace se tyto síly znovu přepočítají. Při formulaci rovnic jsem vycházel z [6].

V programu si může uživatel vybrat typ závodního auta. Typ se volí při přihlášení klienta-řidiče do závodu. Parametry jednotlivých vozů jsou definovány v textových souborech v adresářích `/client/cars` a `/server/cars`. První adresář se použije pro simulaci v testovacím režimu a pro rozměry aut v grafice programu, druhý při spuštění závodu na serveru. Je možné tyto soubory měnit a přidávat nová auta. Změny parametrů se projeví při připojení řidiče do závodu.

4.5 Povrchy tratě

Modely aut se pohybují po terénu reprezentovaném trojúhelníkovou sítí. Tato síť se skládá z trojúhelníků o hraně 20m, které tvoří rovnou plochu velikosti jednoho kilometru čtverečního. Okraje plochy jsou ohraničené bariérou, aby závodníci nemohli propadnout do prázdného prostoru.

Chování auta na určitém typu povrchu ovlivňuje součinitel tření mezi pneumatikou a terénem. V JBulletu jej lze nastavit pro každé kolo parametrem `frictionSlip`. Také

Povrch	Textura	Koeficient
Asfalt	asphalt	1,0
Beton	concrete	1,0
Hlína	dirt	0,65
Šotolina	gravel	0,6
Písek	sand	0,60
Tráva	grass	0,5
Sníh	snow	04
Led	ice	0,2

Tabulka 1: Povrchy tratě

je možné nastavit tření pro model terénu a každé těleso, na chování auta to však nemá žádný vliv. Koeficient pneumatik se mění dynamicky podle povrchu trati, po němž auto jede. Pro zjednodušení je vždy stejný pro všechna kola a určuje jej pozice středu auta. Hodnota tohoto součinitele tření se předává neuronové síti. V tabulce 1 jsou uvedeny koeficienty pro různé povrchy. Hodnoty pocházejí z [7], ale jsou upravené pro potřeby aplikace. Některá auta mají bonusy, které se přičtou k součiniteli tření při jízdě na povrchu určitého typu. Program rozlišuje `asphaltBonus` pro pevné povrchy tj. asfalt a beton nebo `offroadBonus` pro nezpevněné povrchy - to jsou všechny ostatní.

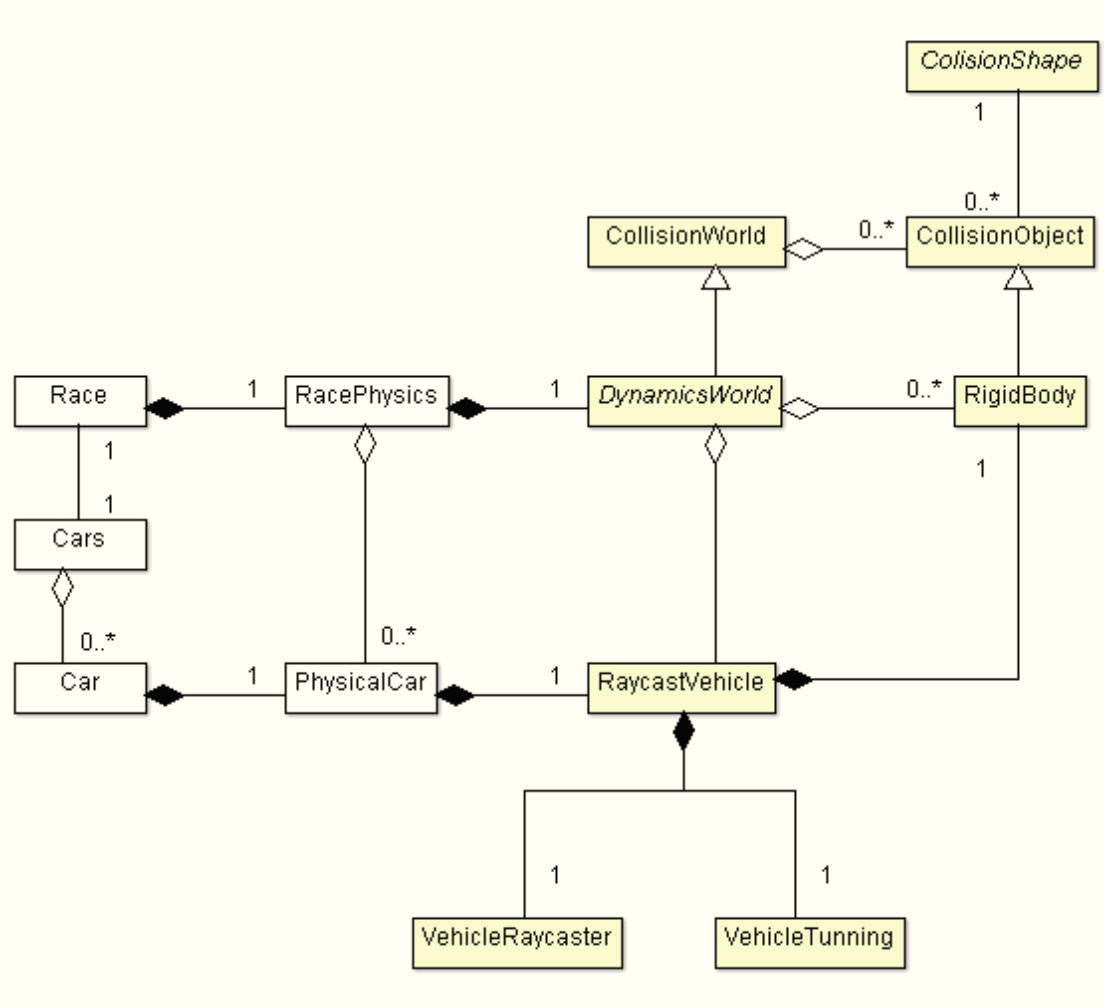
Součinitel tření se předává neuronové síti jako vstup s názvem `friction`. Nabývá hodnot od 0,0 (žádné tření) do 1,0 (asfalt/beton).

Na povrchu s menším třením se může vozidlo snadno dostat do smyku. Zda je auto ve smyku lze zjistit z fyzikálního modelu z proměnné `skidInfo`. Hodnota 0,0 značí, že pneumatika klouže po povrchu, hodnota 1,0 znamená, že pneumatika má trakci. Tato proměnná má jinou hodnotu pro každé kolo. Neuronové síti je předáván vstup `skid`, který je průměrem pro všechna kola. Typické hodnoty vstupu sítě jsou:

- 0,0 – Všechna kola jsou ve smyku.
- 0,5 – Jedna náprava je ve smyku.
- 1,0 – Všechna kola mají trakci.

4.6 Zabudování fyzikálního modelu do programu

Jak funguje propojení fyzikálního s logikou aplikace je vidět na obr. 8. Ke každému závodu `Race` je přiřazena instance třídy `RacePhysics`, která slouží jako rozhranní mezi logikou závodu a fyzikálním modelem reprezentovaným třídou `DynamicsWorld`. `RacePhysics` obsahuje metody pro vytváření modelů objektů a vozidel a posouvání simulace. Terén, objekty na trati a karoserie vozidel jsou reprezentovány instancemi třídy `RigidBody`. Dynamiku vozidla simuluje třída `RaycastVehicle`. Třída `PhysicalCar` umožňuje načíst parametry vozidla ze souboru, poté ovládá jeho model a získává z něj data. Třída `Car` slouží k převodu získaných dat do logiky programu a předání těchto dat řidičům a prohlížečům.



Obrázek 8: Interakce programu s fyzikálním modelem

V průběhu závodu je nejdřív nastaveno řízení každého auta podle údajů od řidiče - brzdový pedál, plynový pedál a natočení volantu. Podle toho se spočítají síly poháněj́ící nebo brzdící auto. Potom je proveden krok simulace. Následně se odečtou data z modelu - souřadnice středu auta, otočení kolem svislé osy, natočení kol, rychlost auta a hodnota `skidInfo` (indikátor smyku) pro každé kolo. Také je nastaven součinitel tření pneumatik podle povrchu, na kterém auto jede. Získaná data se převedou na vstupy sítě a pošlou klientům.

5 Překážky na trati

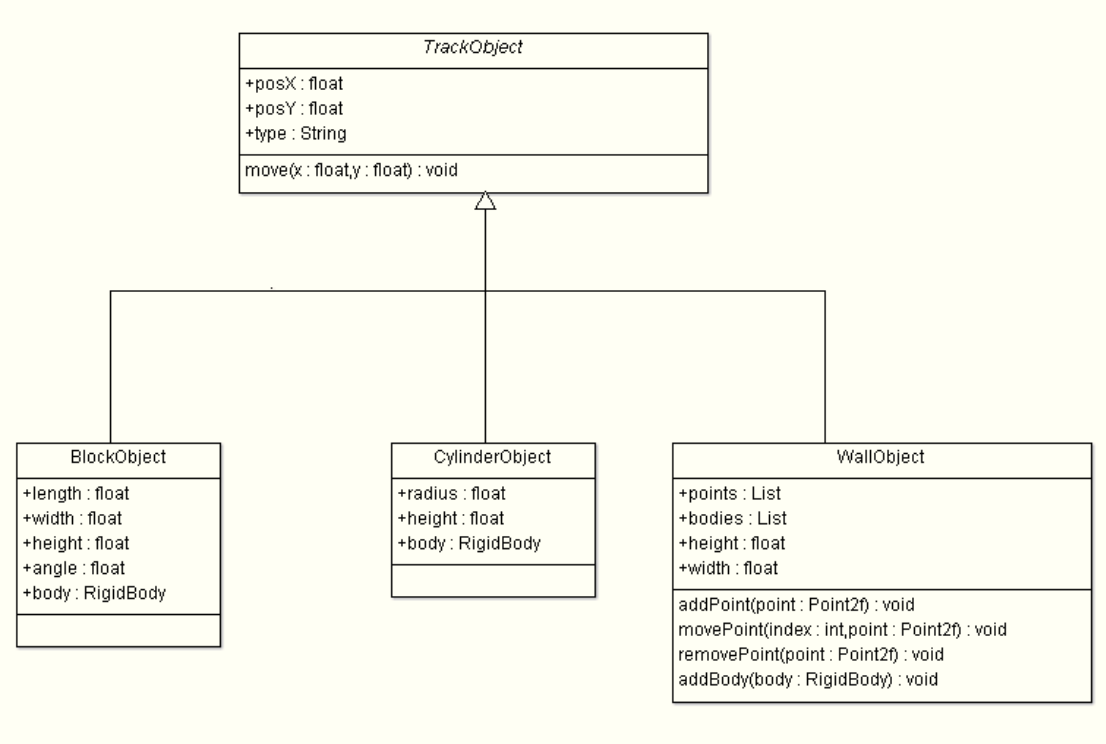
5.1 Typy objektů

Protože fyzikální knihovna JBullet zajišťuje i detekci kolizí, je možné na trať přidat překážky, kterým se auto bude vyhýbat. V aplikaci má každý objekt představující překážku svoji třídu, která určuje tvar kolizního modelu, a typ - ten specifikuje jeho grafickou reprezentaci (jakou použít bitmapu). Bitmapa může mít jiné rozměry než kolizní model, proto je možné nechat v programu zobrazit jeho obrys. Všechny objekty stejného typu mají také shodné rozměry. Do programu jsem implementoval jsem tři třídy objektů:

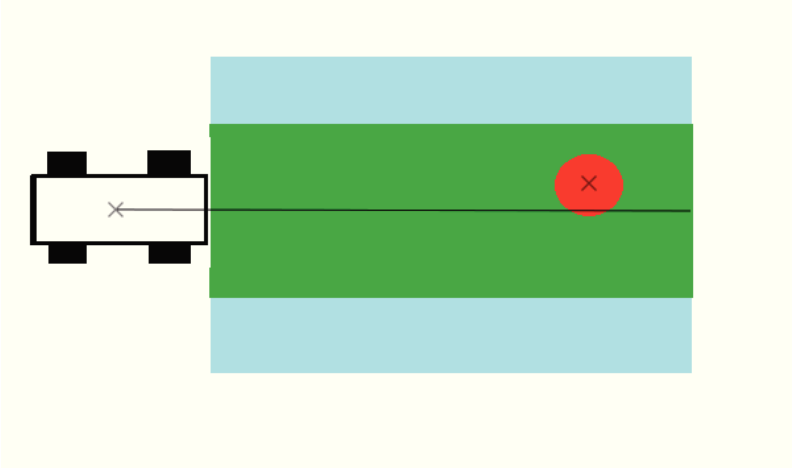
- `blockObject` – Kvádr definovaný výškou, šířkou a délkou a úhlem otočení kolem svislé osy.
- `cylinderObject` – Válec definovaný poloměrem a výškou.
- `wallObject` – Zeď skládající se z více spojených kvádrů. Je definována výškou, šířkou a seznamem bodů, kterými prochází.

Hierarchie objektů je znázorněna na obr. 9. Všechny objekty dědí ze třídy `TrackObject`, která obsahuje typ objektu a souřadnice středu, resp. prvního bodu u zdi. Třídy pro kvádr a kužel specifikují rozměry objektů, u zdi je přidán ještě seznam bodů. V případě, že je vytvořen závod s fyzikálním modelem, je každému objeku přiřazeno statické pevné těleso s příslušným tvarem (zeď se může skládat z více těles).

Rozměry jednotlivých typů objektů jsou definovány v textových souborech v adresáři `/client/objects`. V jednom textovém souboru může být více objektů. Ve stejném adresáři jsou i bitmapové obrázky znázorňující tyto objekty. Název obrázku je shodný s typem objektu. Velikost obrázku v grafice aplikace je dána také v příslušném textovém souboru. Textové soubory je možné editovat, provedené změny se projeví při spuštění programu. Šířku a výšku zdí je možné nastavit přímo v programu. Rozměry objektů umístěných na trati už nelze měnit.



Obrázek 9: Typy objektů



Obrázek 10: Senzory zjišťující překážky

5.2 Detekce překážek

Auto zjišťuje překážky před ním pomocí dvou sezorů obdélníkového tvaru. Ty jsou znázorněny na obrázku 10. Šířka senzorů je závislá na rozměrech auta. Šírka prvního (zeleného) senzoru je dvojnásobek a šířka druhého (modrého) čtyřnásobek šířky karoserie. Dosah senzorů je závislý na rychlosti vozidla, minimální dosah je 20 metrů. Senzory je možné zobrazit v grafice aplikace.

Pokud je překážka v zeleném obdélníku, hrozí srážka s překážkou a je třeba se jí vyhnout. Neuronová síť by měla otočit auto tak, aby se překážka dostala do modrého obdélníku. Pokud se objekt nachází v modrém obdélníku, už srážka nehrozí, ale vozidlo by se nemělo otáčet zpátky směrem k překážce.

Při zjištění překážky senzorem se neuronové síti předává pozice geometrického středu překážky relativně k vodící čáře. Vstupy sítě pro jednotlivé senzory mají názvy `sensor1` a `sensor2`. Sensor vždy předá jen pozici překážky, která je vozidlu nejblíže. Hodnoty vstupů jsou následovné:

- 0.0 – Nalezena překážka se středem nalevo od středové čáry.
- 0.5 – Sensor nezjistil překážku.
- 1.0 – Nalezena překážka se středem napravo od středové čáry.

Tento systém senzorů je velmi jednoduchý a je vhodný k vyhýbání se osamoceným překážkám. Více objektů vedle sebe může způsobit problém, kdy auto neví, které překážce se dříve vyhnout.

6 Závody

6.1 Checkpointy

Projetí trati je kontrolováno pomocí checkpointů. V předešlé verzi programu byly checkpointy i způsob jejich vytváření uživateli úplně skryty. Také nebylo jasné, jestli závodník kontrolním bodem projel nebo ne. Nyní se checkpointy vytvářejí v řídících bodech křivky a návrhář tratě může určit, ve kterých bodech budou umístěny. Na trati jsou checkpointy vyznačeny modrou čarou a start čarou červenou. Neuronové síti se předává směr ke středu dalšího kontrolního bodu:

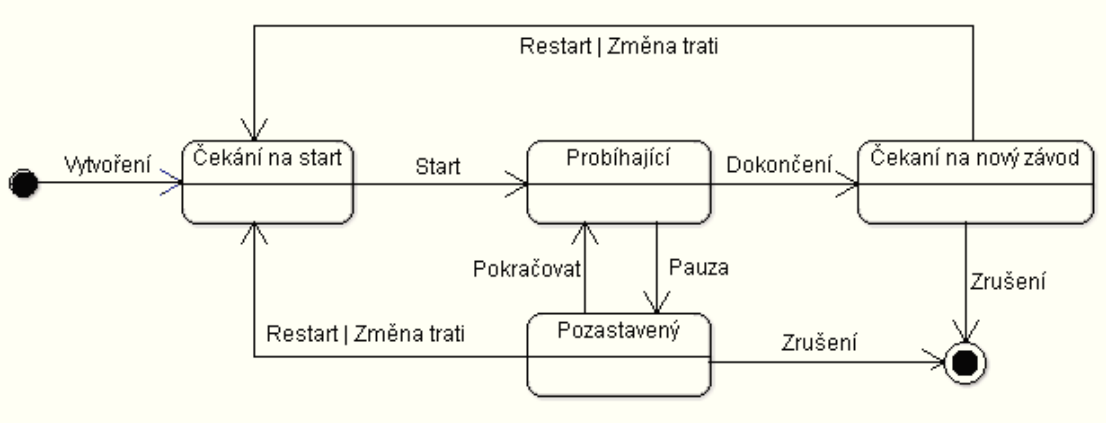
- 0.0 až 0.5 – Checkpoint je nalevo od středu auta (-180° až 0°).
- 0.5 až 1.0 – Checkpoint je nalevo od středu auta (0° až 180°).

Směr k následujícímu checkpointu je možné zobrazit v grafice programu jako modrou šipku. Kontrola projetí checkpointu má určitou toleranci, takže je uznáno i projetí těsně vedle něj.

6.2 Průběh závodu

V původním programu bylo možné závod pouze vytvořit, spustit a ukončit. Po skončení závodu se navíc všichni závodníci odpojili a musel se vytvořit nový závod. V nové verzi aplikace je implementován propracovanější systém organizace závodů. Jak tento systém funguje bude popsáno v následujících bodech:

1. Uživatel přes prohlížeč vytvoří nový závod. Určí parametry závodu: trať, počet kol a zda zapnout kolize mezi auty. Může také vytvořit tzv. šampionát. Šampionát je série o dvou až deseti závodech, kdy se sčítají body získané v jednotlivých závodech a na konci je určen celkový vítěz. Body se jezdcům přičítají jen při dokončení závodu. Uživatel, který závod vytvořil, se automaticky stává jeho pořadatelem.
2. Následně se mohou připojovat diváci (ostatní prohlížeče) a závodníci (programy s neuronovou sítí). Diváci průběh závodu nemohou ovlivňovat.
3. Až se sejde dost závodníků, může pořadatel závod odstartovat. Pokud jsou kolize vypnuty, všechny vozy startují ze stejného místa. Pokud jsou zapnuty, seřadí se za sebou ve startovním poli.
4. V průběhu závodu jej může pořadatel pozastavit.
5. Závod je řádně ukončen, pokud všichni jezdci dojedou do cíle nebo vyprší časový limit. Časový limit začne běžet po projetí prvního závodníka cílem. Je tedy nutné, aby závod dokončil alespoň jeden jezdec. Po skončení závodu dojde k vyhodnocení a uložení výsledků.



Obrázek 11: Stavy závodu

6. Potom může pořadatel vybrat novou trať nebo spustit závod znovu se stejným nastavením. Závodníci i diváci zůstanou připojení. Změnit trať nebo restartovat závod lze i po jeho pozastavení (např. pokud žádný jezdec není schopen dojet do cíle). Šampionát se posune do dalšího závodu, jen pokud byl řádně ukončen ten předchozí.

7. Závod lze úplně ukončit odpojením pořadadatele.

Stavový diagram závodu je zobrazen na obr. 11

6.3 Přihlašování a systém bodování

Studenti mohou přihlašovat své klienty-řidiče přes LDAP zadáním svého loginu a hesla. Závodní server se připojí přes protokol SSL na školní server ldap.vsb.cz a ověří jejich totožnost. V případě úspěšného přihlášení se studentovi ukládají výsledky po skončení závodu.

Do výsledků se ukládá nejlepší čas studenta pro každou trať. U času je uveden typ auta, kterým rekordu dosáhl. Dále se ukládají statistiky bodovaných závodů. Závod je bodovaný, pokud se ho zúčastní aspoň 3 závodníci. Na konci dostanou všichni účastníci tolik bodů, kolik porazili ostatních řidičů. Výherce navíc dostane 2 bonus body a druhý v pořadí 1 bonus bod. Například vítěz v závodě čtyř jezdců dostane 3 body + 2 bonus body. Jezdci na stupních vítězů (první až třetí) dostanou navíc medaile. Jezdci, kteří nedojedou do cíle žádné body nezískají.

Výsledky se ukládají do databáze. Databáze by měla běžet na stejném počítači jako závodní server. Server používá pro přístup do databáze heslo, které je možné nastavit příkazem při spuštění serveru. Klienti mohou z databáze pouze načítat data.

7 Uživatelské rozhraní

GUI bylo převzato z původního programu, je však upraveno pro snadnější použití a rozšířeno o nové funkce. Panely trať a editor byly sloučeny do jednoho, takže je možné načítat a ukládat trať přímo v editoru. Byl přidán nový panel žebříčky, kde se nachází výsledky závodníků. Ve všech dialozích s výběrem trati se zobrazují náhledy. Aplikace si nyní pamatuje poslední hodnoty parametrů zadaných uživatelem při vytváření závodu a není třeba je příště zadávat znovu. Nastavení programu se ukládá pomocí API preferences.

7.1 Editor tratí

Editor tratí byl podstatně vylepšen. Základem editoru je stále modelování tratě pomocí Beziérových křivek, nyní ale nabízí více možností. Například možnost měnit typy povrchu: při vytvoření trati uživatel zadá, jaký bude základní typ povrchu vozovky a povrch okolí. Lze měnit i šířku závodní dráhy. V editoru je pak možné každý úsek trati „obarvit“ zvoleným typem povrchu. Okolí tratě je tvořeno jedním typem povrchu, který lze změnit ve vlastnostech.

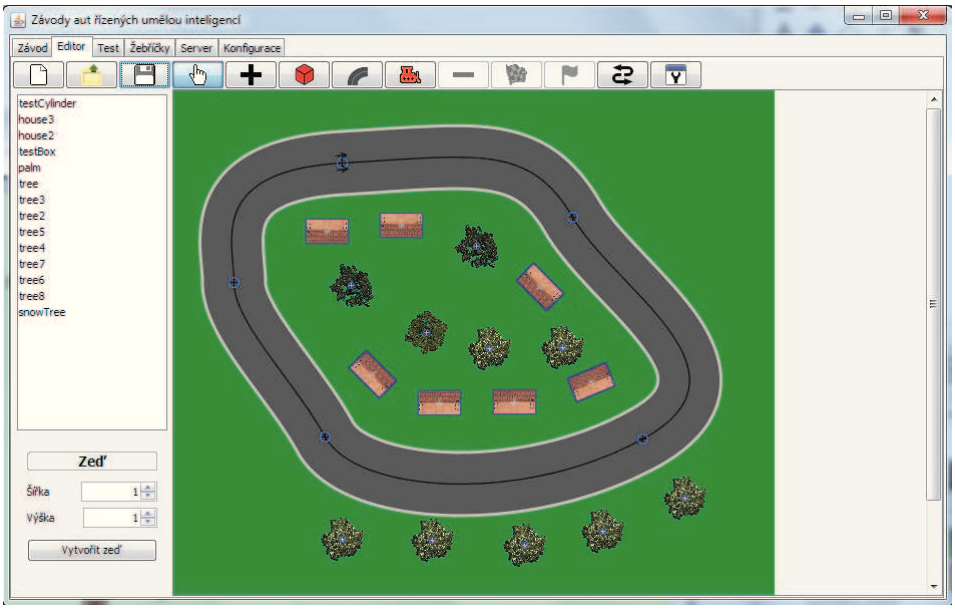
Na trať je také možné přidávat různé objekty, které si může uživatel vybrat z postranního menu. S předměty obdélníkového půdorysu lze před umístěním rotovat pomocí kolečka myši. U objektů je modře vyznačen tvar kolizního modelu. Speciálním typem objektu jsou zdi, u kterých uživatel postupně zadává body, kterými budou procházet. Po umístění objektů je možné přesouvat je zachycením objektu a tažením myší.

V editoru funguje oddalování a přibližování pohledu (zoom) kolečkem myši. Při změně měřítka se přepočítají všechny objekty na nový počet pixelů. Maximální velikost editovatelné plochy je stanovena na 1000x1000 metrů.

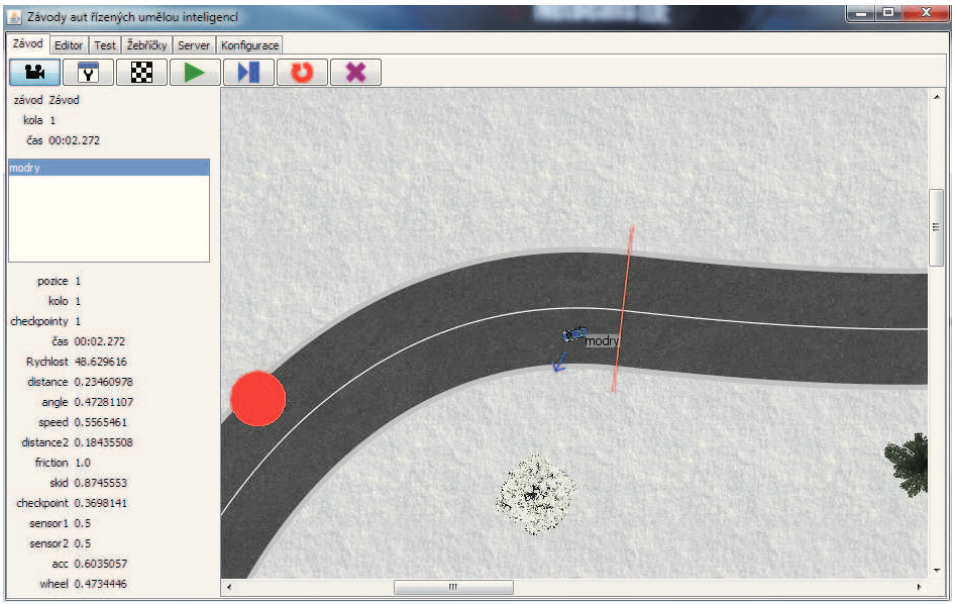
7.2 Grafika závodu

K vykreslování dvojrozměrného zobrazení závodu používá program třídu Graphics2D. V původní verzi byla jednoduchá grafika založená na geometrických tvarech. V nové verzi je většina objektů reprezentována rastrovými obrázky - tzv. sprity. Sprity se nacházejí v adresářích `/client/cars` a `/client/objects`. Velikost spritů v grafice programu je dána jejich rozměry v textových souborech. Trať je vykreslena pomocí textur, které představují různé povrchy. Textury pro povrchy jsou umístěny v adresáři `/client/surfaces`. Je také možné textury vypnout a vykreslovat povrchy jako jednobarevné plochy. Trať a objekty jsou zobrazovány v základním měřítku 1 metr = 5 pixelů. Auta mají oproti trati pixelů čtyřikát více a jsou tak vykreslena s většími detaily. Stejně jako v editoru lze i při závodě pohled přibližovat a oddalovat.

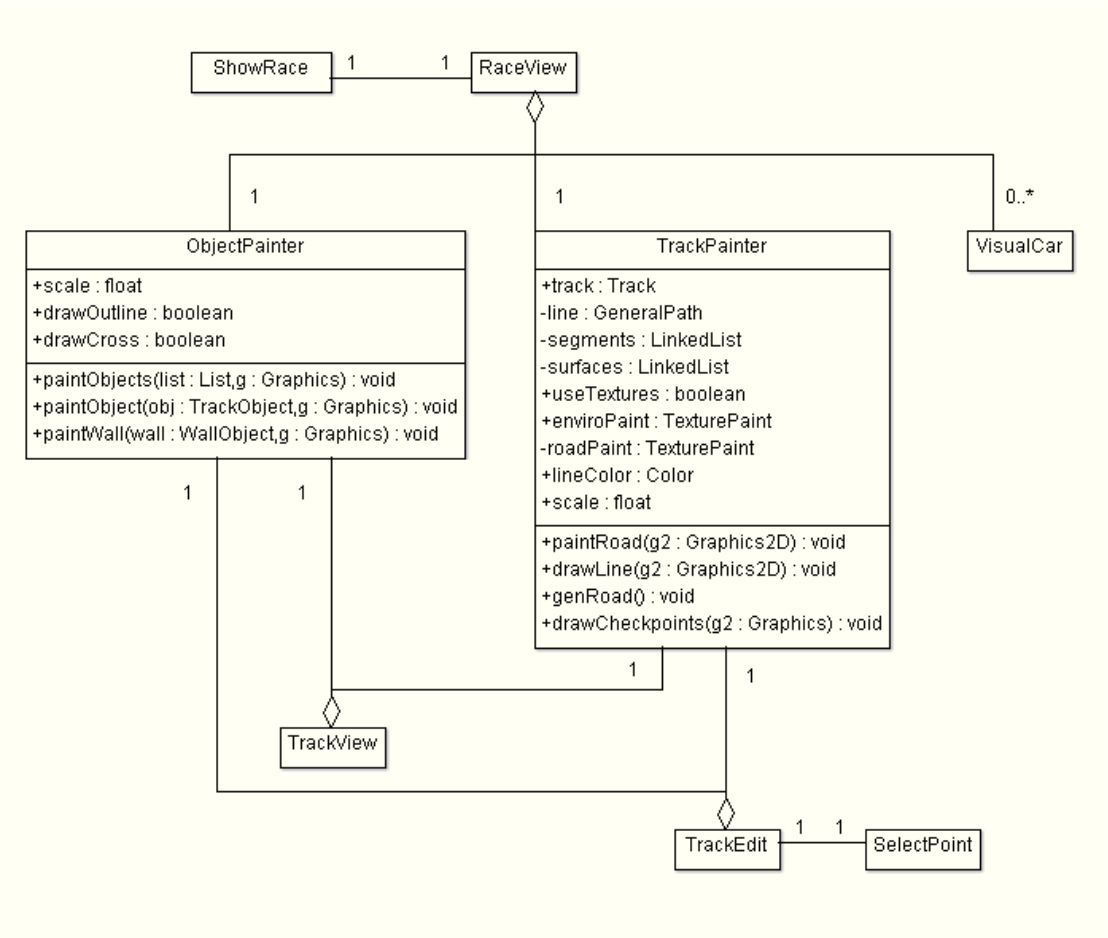
Na obr. 14 je třídní diagram pro grafiku aplikace. `ShowRace` komunikuje se serverem a získává od něj potřebná data. `RaceView` zobrazuje grafiku závodu. K tomu používá třídy `TrackPainter`, `ObjectPainter` a `VisualCar`. `TrackPainter` slouží k vykreslení cesty, pozadí a checkpointů. `ObjectPainter` vykresluje všechny objekty na trati. `VisualCar` vykresluje určité auto v závodě a uchovává jeho parametry obdržené od



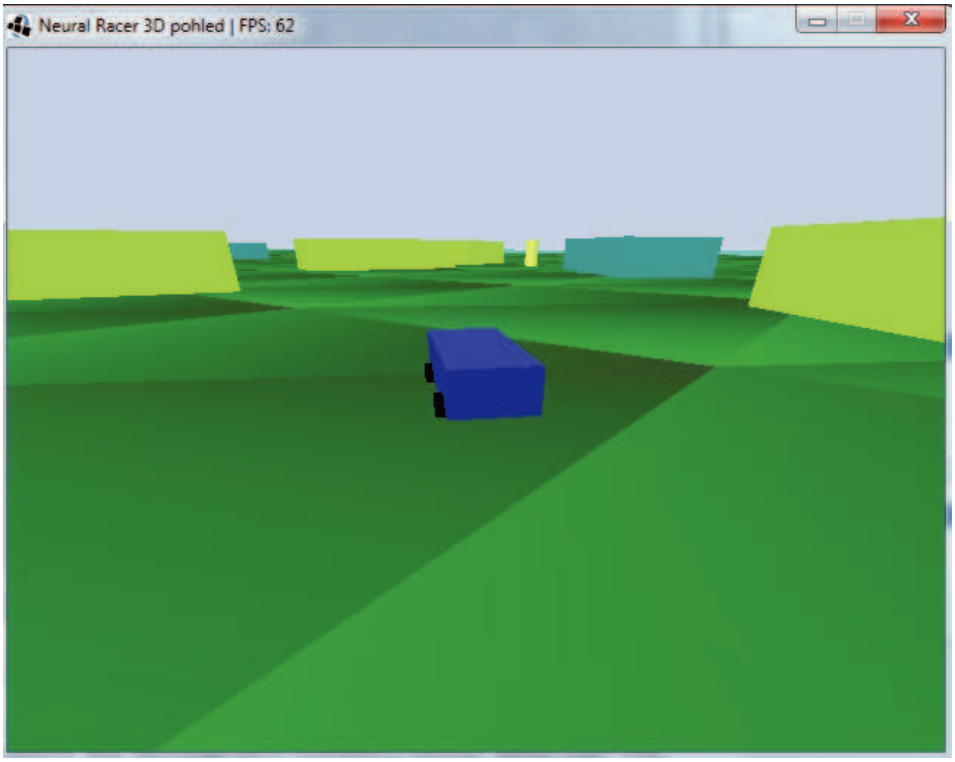
Obrázek 12: Editor tratě



Obrázek 13: Závod



Obrázek 14: Grafika aplikace



Obrázek 15: 3D zobrazení

serveru. `TrackView` slouží k zobrazení náhledů v dialogových oknech s výběrem trati. `TrackEdit` zobrazuje trať v editoru a umožňuje její editaci. `SelectPoint` vykresluje vektory křivky pro vybraný bod.

Uživatel může v nastavení zobrazení zvolit, které informace budou prezentovány v grafice programu. Lze zobrazit obrysy kolizních modelů, senzory zjišťující překážky, směr k dalšímu kontrolnímu bodu, jména jezdců a zvýraznit kola aut. Nastavení je možné měnit v postraní nabidce nebo klávesovými zkratkami.

V aplikaci jsou použity textury z webu [8] a sprity stromů ze stránky [9]. Sprity aut jsou vytvořeny z fotografií s pohledem seshora.

7.3 Testovací režim

Testovací režim slouží k simulaci závodů lokálně, aniž by se uživatel musel připojit k serveru. V původním programu bylo možné připojit pouze jednoho klienta-řidiče. V nové verzi jsem vytvořil zjednodušený testovací závodní server, ke kterému se může kdykoliv připojit i více řidičů. Testovací server se spustí lokálně po stisknutí tlačítka uživatelem. Samotné závody probíhají stejně, pouze mají neomezený počet kol a neprobíhá vyhodnocení a ukládání výsledků. Je možné editovat trať a objekty při probíhajícím testu.

V testovacím režimu je možné zobrazit závod ve 3D (obr. 15). K hardwarové akceleraci grafiky přes OpenGL je použita knihovna LWJGL. Samotné 3D zobrazení je upravenou

verzí dema `vehicle` z JBulletu a ukazuje kolizní modely fyzikálních těles. Výjimkou jsou kola auta, která nemají kolizní model (viz kapitola 4.3), ale jsou vykreslena z estetických důvodů. Kamera je v 3D pohledu umístěna za vozidlem a je možné přepínat se mezi jednotlivými závodníky.

8 Závěr

Na konec shrnu provedená vylepšení programu. Odstranil jsem některé nedostatky původního systému a přidal některé nové funkce. Velkou částí práce bylo zabudování fyzikální knihovny do simulace závodů. Vozidla se nyní chovají reálněji a je trochu těžší je řídit. Také se zvýšil počet informací, které je možné předávat jako vstupy neuronové síti. V editoru lze vytvářet rozmanitější závodní tratě s překážkami. Do programu je možné přidávat další objekty a závodní auta editací souborů. Nový systém organizace závodů a vylepšené GUI umožňuje snažší práci se závodním serverem. Výsledky jezdců v závodech se ukládají do databáze. Zdokonalená grafika prohlížeče by měla učinit sledování závodu zajímavější.

Program je stále možné vylepšit o další funkce. Například by bylo vhodné zdokonalit systém senzorů detekujících překážky na trati a ostatní auta. Překážky by mohly být i dynamické, to znamená, že by se mohly pohybovat a reagovat na náraz auta. Užitečný by byl prohlížeč typů automobilů, kde by se zobrazovaly jejich technické parametry. Také by bylo možné předělat grafiku plně do 3D a upravit editor, aby mohl vytvářet plně trojrozměrné tratě.

Věřím, že provedená rozšíření učiní práci se systémem mnohem zajímavější a uživatelé s ním budou spokojeni.

9 Reference

[1] VONDRÁK, Ivo. *Neuronové sítě.* Ostrava : VŠB - TU Ostrava, 1995. 56 s. Dostupné z WWW: http://vondrak.cs.vsb.cz/download/Neuronove_site.pdf.

[2] KRUCINA, Marian. Závody aut řízených umělou inteligencí. [online]. 7.5.2008 [cit. 2012-05-04]. Dostupné z: http://www.cs.vsb.cz/jezek/vyuka/ns/download/text.pdf

[3] DVOŘÁK, Martin. JBullet - Java port of Bullet Physics Library [online]. [cit. 2012-05-04]. Dostupné z: http://jbullet.advel.cz/

[4] COUMANS, Erwin. Bullet 2.76 Physics SDK Manual. [online]. 25.2.2010 [cit. 2012-05-04]. Dostupné z: http://www.bulletphysics.com/ftp/pub/test/physics/Bullet_User_Manual.pdf

[5] MADOCK, Kester. Vehicle Simulation With Bullet. [online]. 16.8.2010 [cit. 2012-05-04]. Dostupné z: https://docs.google.com/document/edit?id=18edpOwtGgCwNyvakS78jxMajCuezotCU_0iezcwiFQc&pli=1

[6] MONSTER, Marco. Car Physics for Games. [online]. Listopad 2003 [cit. 2012-05-04]. Dostupné z: http://www.asawicki.info/Mirror/Car\%20Physics\%20for\%20Games/Car\%20Physics\%20for\%20Games.html

[7] Tire friction and rolling resistance coefficients. [online]. [cit. 2012-05-04]. Dostupné z: http://hpwizard.com/tire-friction-coefficient.html

[8] CG Textures [online]. [cit. 2012-05-04]. Dostupné z: http://www.cgtextures.com/

[9] Gamemakerfoundry: Tree sprite collection vol.1. [online]. [cit. 2012-05-04]. Dostupné z: http://gamemakerfoundry.com/2011/02/25/tree-sprite-collection-vol-1/

A Uživatelský manuál

A.1 Režimy aplikace

Systém pro závodění aut řízených umělou inteligencí je zaměřen na závodění aut v prostředí internetu. Systém lze spouštět ve dvou režimech:

- Režim s GUI – určen pro spouštění na uživatelských stanicích.
- Režim bez GUI – určen pro spouštění jako závodní server.

Oba režimy se spouští souborem `neurace.jar`. V umístění programu by se měly nacházet adresáře `client` a `server`, které obsahují data aplikace. Režim bez GUI potřebuje k běhu pouze data z adresáře `server`.

A.2 Požadavky

Pro běh systému musí počítač splňovat následující minimum:

- CPU 1.6GHz
- 512MB RAM
- 20MB volného místa na disku
- JRE min. verze 1.7 – možné stáhnout ze stránky http://www.java.com/
- Síťová karta pro připojení k serveru
- pro režim s GUI:
  - monitor s rozlišením 800x600 nebo vyšším
  - klávesnice a myš

A.3 Server

Režim bez GUI je určen pro nasazení na server. Je možné jej vzdáleně spravovat přes příkazovou řádku. Server se spouští následovně:

```
java -jar neurace.jar server [nepovinné parametry]
```

Nepovinné parametry jsou:

- `-port <cislo portu>` – TCP port, na kterém program naslouchá, číslo portu je v rozmezí 0 až 65535.
- `-dir <cesta k adresari>` – cesta k adresáři, ve kterém jsou data serveru. Aplikace by měla mít právo čtení a zápisu do zvoleného adresáře a měl by obsahovat podadresáře `cars` a `tracks`. V základním nastavení použije adresář `server` v umístění programu.

- `-ldap on` – zapne přihlašování přes LDAP. Zapnuto v zákl. nastavení.
- `-ldap off` – vypne přihlašování přes LDAP
- `-ups <počet kroků za sekundu>` – nastaví počet kroků simulace za sekundu (10 až 60).
- `-dbport <cislo portu>` – port který používá server k připojení k databázi.
- `-dbpass <nové heslo>` – změní heslo pro přístup do databáze.
- `-help` – zobrazí seznam parametrů (nespouští server).

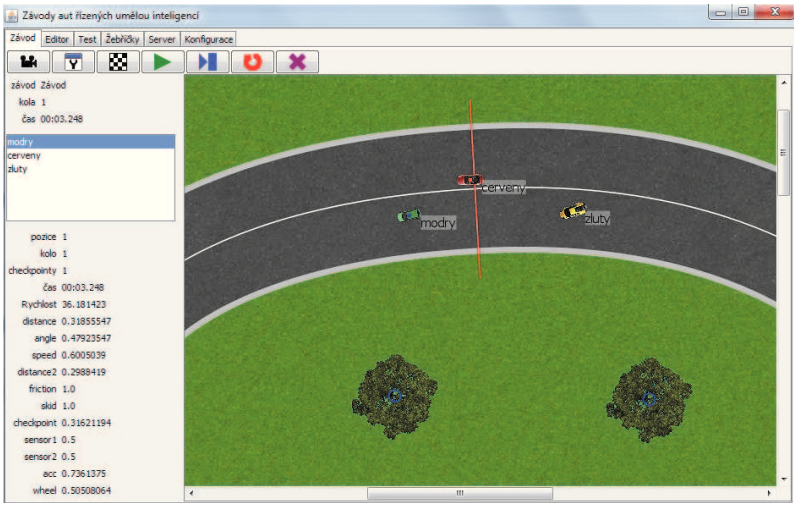
Server si nastavení pamatuje, použije jej i při dalším spuštění. Na určený port se mohou připojovat klienti, kteří budou pracovat s tratěmi a hlavně pořádat závody. Při použití LDAP přihlašování by měl mít server možnost spojit se s ldap.vsb.cz. Běžící server je možné zastavit příkazem `s` (stop).

Pro ukládání výsledků závodů je třeba, aby na počítači se závodním serverem běžela databáze Java DB (Apache derby). K programu je přiložen konfigurační soubor databáze `derby.properties`. Server se k databázi připojuje jako uživatel `server` s heslem nastaveným příkazem `-dbpass` a měl by mít oprávnění měnit data. Prohlížeč se přihlašuje jako `client` s heslem `client` a měl by mít povoleno číst z databáze.

A.4 GUI

Režim s GUI je určen pro užvatelské stanice. Tento režim zahrnuje možnosti celého systému včetně konfigurace a spouštění závodního serveru. Systém se spouští příkazem `java -jar neurace.jar` nebo dvojklikem na soubor `neurace.jar` myší. Pro připojení k závodnímu serveru je nutné nejdřív nakofigurovvat připojení v panelu konfigurace. Celý systém běží v jednom okně a jednotlivé části jsou rozdělené do panelů podle funkce:

- Závod – vytváření a sledování závodů na serveru.
- Editor – načtení a editace vybrané tratě.
- Test – testování klienta na načtené trati bez nutnosti připojovat se k serveru.
- Žebříčky – nejlepší časy na jednotlivých tratích a bodování jezdců.
- Server – konfigurace a ovládání serveru.
- Konfigurace – konfigurace klienta.



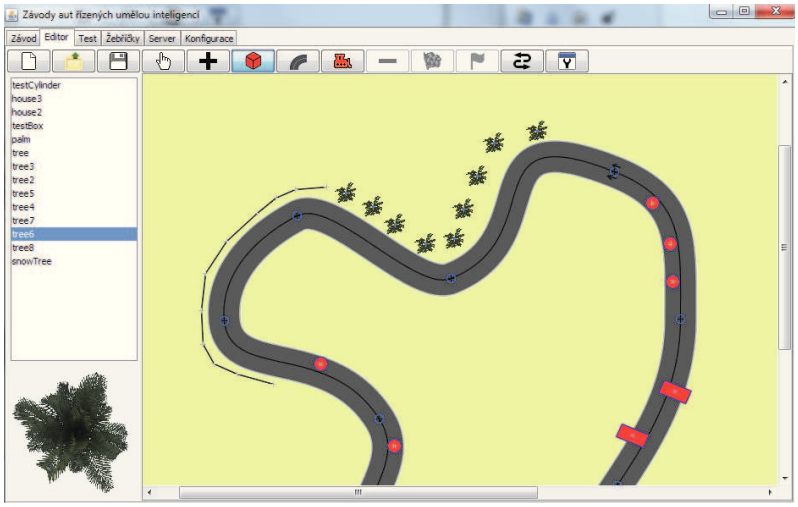
Obrázek 16: Panel Závod

A.4.1 Závod

Prostřednictvím panelu Závod (obrázek 16) lze řídit a prohlížet závod probíhající na závodním serveru. Závodní server může běžet na jiném počítači v síti. Pro připojení na server je nutné specifikovat název počítače a port, na kterém server běží, na panelu Konfigurace. Na tomto panelu můžete vytvářet nové závody, ty spouštět a pozastavovat. Pokud vytvoříte závod, stáváte se jeho pořadatelem. Odpojením pořadatele závod končí. U nového závodu můžete vybrat trať, počet kol a zapnout kolize mezi auty. Také je možné vytvořit tzv. šampionát. Šampionát je série o dvou až deseti závodech, kdy se získané body sčítají a na konci je určen celkový vítěz. Body se přičítají jen při dokončení závodu. Kolik závodů ještě zbývá do konce šampionátu, je možné vidět v postranním panelu.

Dále je možné připojit se jako divák a pozorovat závod běžící na závodním serveru. Pro tyto účely slouží následující tlačítka:

- Nový závod – vytvoří nový závod a připojí se k němu jako pořadatel.
- Připojit se – připojí se k existujícímu závodu jako divák.
- Sledovat vozidlo – kamera bude během závodu sledovat vybrané vozidlo.
- Možnosti zobrazení – otevře nabídku možností zobrazení závodu. Změny jsou provedeny okamžitě, je možné uložit nastavení i pro další spuštění programu.
- Změnit trať – otevře okno pro výběr trati. Potvrzením dialogu se spustí závod na jiné trati se stejnými závodníky. Lze měnit i počet kol a nastavení kolizí.
- Start – odstartuje závod.
- Pauza – pozastaví závod.



Obrázek 17: Panel Editor tratě

- Restart – vrátí závodníky zpět na start.
- Krok – provede jenom jeden krok závodu.
- Odpojit – odpojí se od závodu. V případě pořadatele ukončí závod.

Možnosti zobrazení:

- Směr k checkpointu (A)– zobrazí šipku směřující k dalšímu kontrolnímu bodu.
- Jména jezdců (N)– zobrazí jména všech jezdců.
- Obrysy objektů (O)– zobrazí obrysy fyzikálního modelu objektů na trati.
- Senzory (S)– zobrazí senzory zjišťující překážky u vybraného auta.
- Textury (T)– použije na povrch trati a pozadí textury.
- Zobrazit kola (W)– zvýrazní kola aut.

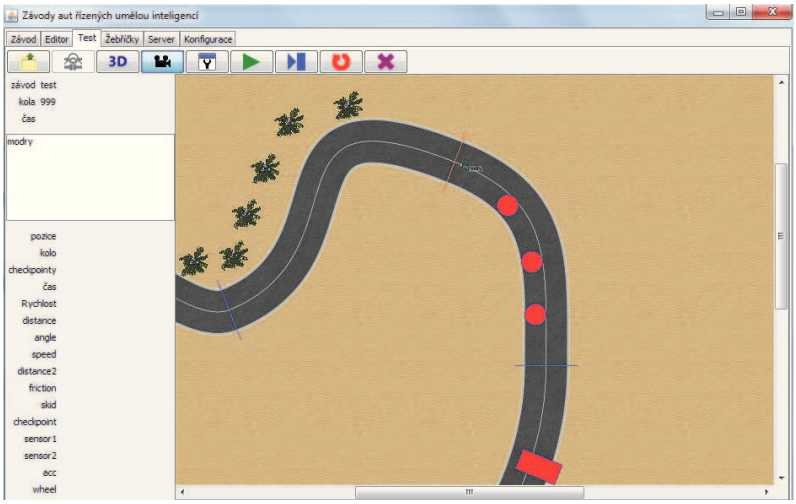
Ovládání:

- Kolečko myši nebo klávesy +/- – přibližování/oddalování pohledu
- F - přepne sledování vozidla
- Šipky – posouvání pohledu do stran (pokud kamera nesleduje vozidlo)
- Tažení myší – posouvání pohledu (pokud kamera nesleduje vozidlo)

A.4.2 Editor tratě

Na panelu Editor tratě (obrázek 17) je umístěn editor tratě. Editace spočívá v manipulaci s body, které tvoří trať. Pokud trať ještě neexistuje (plocha je prázdná), je nutno ji vytvořit přidáváním bodů. Nový bod se vytvoří stiskem levého tlačítka myši a následovným tažením se nastavují řídící body pro vstupní a výstupní křivku. Trať musí obsahovat minimálně dva body. Pro dokončení stiskněte pravé tlačítko myši nebo tlačítko Označ bod. Pokud je trať uzavřená, je možno ji dál upravovat přesouváním bodů a změnou jejich vektorů, nebo přidáváním a odstraňováním bodů. Editor má následující ovládací prvky:

- Nová trať – zobrazí dialogové okno s vlastnostmi nové tratě. Po odsouhlasení dialogového okna se zobrazí prázdné pole, na které je možno v editoru tratí vytvořit trať.
- Načíst trať – rozvine nabídku možností pro načtení tratě:
  - Načíst z disku – zobrazí standardní dialog pro výběr souboru. Po výběru je trať načtena.
  - Načíst ze serveru – zobrazí seznam tratí uložených na serveru. Po výběru je trať načtena.
- Uložit – rozvine nabídku možností pro uložení tratě:
  - Uložit na disk – uloží trať do souboru.
  - Uložit na disk jako – umožní výběr souboru, do něhož potom uloží trať.
  - Uložit na server – uloží trať na server.
  - Uložit na server jako – umožní výběr názvu tratě, pod kterým ho uloží na server.
- Označ a přesuň – přepne editor do režimu označování a přesouvání bodů a objektů. Objekty je možné je přesouvat zachycením za středový křížek a tažením myší. Obdélníkové objekty lze otáčet kolečkem myši za přidržení levého tlačítka. Do tohoto režimu je možné se kdykoli vrátit pravým kliknutím myši.
- Nový bod – přepne editor do režimu vkládání bodů. Tažením myši lze nastavit vektory pro vstupní a výstupní křivku. Body jsou označeny černými křížky.
- Přidat objekty – přepne editor do režimu vkládání objektů. Objekt je možné vybrat z postranního menu. Objekty obdélníkového půdorysu lze otáčet kolečkem myši. Objekt umístíte levým tlačítkem. Modrý obrys znázorňuje kolizní model objektu. Při umisťování zdi/bariéry se postupně vkládají body, kterými má zeď procházet. Vkládání bodů lze ukončit pravým tlačítkem.
- Povrch cesty – přepne editor do režimu změny povrchu cesty. Typ povrchu si můžete vybrat z postranního menu. Poté je možné obarvit část trati kliknutím na některý bod.



Obrázek 18: Panel Test

- Smazat objekty – přepne editor do režimu odstraňování objektů. Objekt odstraníte kliknutím na středový křížek.
- Odstraň – odstraní vybraný bod.
- Nastavit první – vybraný bod určí jako místo startu.
- Checkpoint – vytvoří nebo zruší checkpoint ve vybraném bodě. Checkpoint označuje modrý kruh.
- Otočit směr – změní směr jízdy.
- Vlastnosti tratě – Zobrazí dialogové okno pro změnu vlastnosti tratě. Ve vlastnostech tratě lze určit velikost plochy, na které je umístěna trať, základní povrch cesty (pouze pro nově přidané body), šířku cesty a povrch okolí trati.

Ovládání:

- Kolečko myši nebo klávesy +/- – přibližování/oddalování pohledu
- Šipky – posouvání pohledu do stran
- Kolečko myši – otáčení objektů obdélníkového tvaru
- Pravé tlačítko – návrat do režimu výběru a přesouvání objektů

A.4.3 Test

Panel Test (obrázek 18) je určen pro jednoduché testování klientů řidičů. Pokud je načtená nebo vytvořená trať, pak je možné aktivovat lokální testovací server a připojit klienty, se

kterými bude komunikovat stejně jako během závodu. Na rozdíl od závodu se uživatel nemusí připojovat k závodnímu serveru a vytvářet závod. Pro funkčnost testu je nutné zvolit port, na který se klienti připojí v nastavení programu. Port se nastavuje na panelu Konfigurace. Testovací jízdu lze přerušit a zase spustit nebo krokovat. Tlačítko odpojit ukončí test odpojí všechny řidiče.

- Načíst trať – rozvine nabídku možností pro načtení tratě:
  - Načíst z disku – zobrazí standardní dialog pro výběr souboru. Po výběru je trať načtena.
  - Načíst ze serveru – zobrazí seznam tratí uložených na serveru. Po výběru je trať načtena.
- Spusť testovací server – umožní připojování řidičů k závodu na načtené trati.
- 3D zobrazení – otevře 3D zobrazení závodu v novém okně.
- Sledovat vozidlo – kamera bude během závodu sledovat vybrané vozidlo.
- Možnosti zobrazení – otevře nabídku možností zobrazení, stejné jako u závodu.
- Start – spustí test
- Pauza – pozastaví test
- Krok – provede jenom jeden krok jízdy.
- Restart – vrátí závodníky zpět na start.
- Odpojit – odpojí řidiče a ukončí test.

Ovládání:

- Kolečko myši nebo [+/-] – přibližování/oddalování pohledu
- F - přepne sledování vozidla
- Šipky – posouvání pohledu do stran (pokud kamera nesleduje vozidlo)
- Tažení myší – posouvání pohledu (pokud kamera nesleduje vozidlo)

Ovládání 3D pohledu:

- C – přepíná kameru mezi auty
- Levé tlačítko myši – postrčení auta
- Pravé tlačítko myši – vystřelení krychle
- Mezerník – skok auta

A.4.4 Žebříčky

Zde si můžete prohlížet výsledky závodů. Najdete tu nejlepší časy pro každou trať a celkový žebříček úspěšnosti jezdců v bodovaných závodech. Závod je bodovaný, pokud se ho zúčastní aspoň 3 závodníci. Na konci dostanou všichni účastníci tolik bodů, kolik soupeřů porazili. Výherce navíc dostane 2 bonus body a druhý v pořadí 1 bonus bod. Například vítěz v závodě čtyř jezdců dostane 3 body + 2 bonus body. Jezdci na stupních vítězů získají navíc medaile. Závodníci, kteří nedojeli do cíle v časovém limitu žádné body nezískávají. Žebříček je seřazen podle dosažených bodů.

A.4.5 Server

Pro snadné pořádání závodů v jakékoliv síti je možné ovládat server přímo z GUI z panelu Server. Takto spuštěný server je stejný jako spuštěný bez GUI. Server běží v rámci jedné aplikace a ukončením této aplikace je automaticky ukončen i server.

Konfigurace serveru se nachází na stejném místě a skládá se ze dvou voleb:

- Adresář s tratěmi – cesta k adresáři, do kterého systém ukládá tratě. Aplikace by měla mít právo čtení a zápisu do zvoleného adresáře a měl by obsahovat podadresáře `cars` a `tracks`. V základním nastavení se použije adresář `server` v umístění programu.
- Port serveru – TCP port, na kterém poslouchá server, číslo 0 – 65535

Server se ovládá následujícími tlačítky:

- Start – spustí server.
- Stop – zastaví server.

A.4.6 Konfigurace

Na panelu Konfigurace se nastavuje konfigurace pro program. Jsou to hodnoty pro spojení na závodní server a pro testy. Konfigurace aplikace má následující položky:

- Server – www nebo ip adresa závodního serveru v síti.
- Port serveru – port na serveru, na kterém naslouchá závodní server, číslo 0 – 65535
- Port pro testy – TCP port, na kterém naslouchá testovací server, číslo 0 – 65535
- Port databáze – TCP port, na kterém se připojuje klient k databázi, číslo 0 – 65535
- Vzhled – nastavuje look and feel aplikace.
- Lokalizace – přepíná jazyk uživatelského rozhranní. Pro provedení změn je třeba restartovat aplikaci.
- Test připojení – otestuje připojení k serveru.

Hodnoty se aplikují v momentě změny. Konfigurace samotného závodního serveru je přímo na panelu Server.

A.5 Komunikace mezi serverem a řidičem

Server komunikuje s klientem-řidičem pomocí textových zpráv na transportní vrstvě TCP/IP. Zpráva se skládá z libovolného počtu řádků. K označení konce řádků se používá znak LF. ASCII kód tohoto znaku je 0x0A. Po zprávě následuje jeden prázdný řádek.

Řádek, pokud není definováno jinak nebo není prázdný, je ve tvaru `klíč:hodnota` nebo jen `hodnota`. Klíč je řetězec složený z velkých a malých znaků anglické abecedy a z číslic. Definuje hodnotu, která je za dvojtečkou. Hodnota je vše od dvojtečky po konec řádku. Hodnoty jsou následujících typů:

- `string` – řetězec v kódování UTF8, může obsahovat všechny znaky mimo konec řádku.
- `int` – celé číslo v desítkové soustavě.
- `float` – reálné číslo v desítkové soustavě.
- `nfloat` – reálné číslo od 0 do 1 včetně v desítkové soustavě.

V popisu se objevuje text `prázdný řádek`. Toto je řádkem nulové délky, to znamená jen znak konce řádu. Jeho účelem je oddělit od sebe různé zprávy nebo jejich části.

Seznam závodů
K získání seznamu závodu běžících na serveru slouží příkaz:
`racelist` – identifikace protokolu
`prázdný řádek` – konec informace

Odpověď serveru je:
`ok` – potvrzení
`prázdný řádek` – oddělení
Potom pokračuje seznam názvů závodů, na každém jeden.
`prázdný řádek` – konec seznamu

Seznam aut
K získání seznamu aut dostupných ve vybraném závodě slouží příkaz:
`carlist` – identifikace protokolu
`race:string` – název závodu
`prázdný řádek` – konec informace

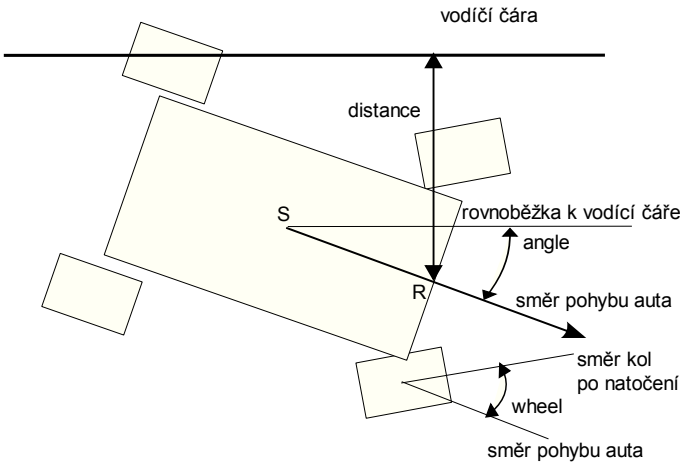
Odpověď serveru je:
`ok` – potvrzení
`prázdný řádek` – oddělení
Potom pokračuje seznam typů aut, na každém jeden.
`prázdný řádek` – konec seznamu

Připojení k závodu
Na začátku komunikace se klient připojí k serveru. Klient pošle severu úvodní informace:
`driver` – identifikace protokolu
`race:string` – název závodu
`driver:string` – jméno řidiče (login)
`password:string` – heslo
`car:string` – typ auta
`color:string` – barva auta ve formátu RGB zapsána v šestnáctkové soustavě, kde každá složka má právě dva znaky.
`prázdný řádek` – konec hlavičky.

Na tuto zprávu může server vrátit jednu z následujících možností a ke každé přidá prázdný řádek.
`error: race not found` – závod nebyl nalezen, konec spojení
`error: driver exists` – řidič toho jména už je v uvedeném závodě, konec spojení
`error: car not available` – vybrané auto není dostupné
`ok` – vše je v pořádku, řidič je připojen do závodu
Při neúspěšném přihlášení přes LDAP vrátí chybovou hlášku z LDAP serveru.

Průběh závodu
Pokud se řidič úspěšně přihlásí k závodu, následuje řízení auta. Tam dochází ke změně rozložení požadavek – odpověď. Server posílá požadavek klientovi a ten mu odpovídá. Při čekání na start závodu posílá server průběžně prázdný řádek, aby zjistil jestli se nějaký jezdec neodpojil. Po startu závodu posílá klientovi informaci o pozici auta a řidič odpoví nastavením směru a rychlosti:
`round` – nový požadavek
`distance:nfloat` – vzdálenost od čáry
`angle:nfloat` – úhel mezi čárou a směrem jízdy
`speed:nfloat` – rychlost auta
`distance2:nfloat` – vzdálenost od čáry za vteřinu (při dodržení stejného směru)
`friction:nfloat` – součinitel tření mezi pneumatikami a povrchem
`skid:nfloat` – určuje zda je auto ve smyku
`checkpoit:nfloat` – směr k dalšímu checkpointu
`sensor1:nfloat` – první senzor
`sensor2:nfloat` – druhý senzor
`prázdný řádek` – konec informace

Odpověď klienta:
`ok` – potvrzení přečtení
`wheel:nfloat` – natočení volantu
`acc:nfloat` – zrychlení
`prázdný řádek` – konec informace



Obrázek 19: Parametry auta

Uvedená komunikace se opakuje, dokud místo `round` server nepošle klientovi `finish` a ukončí spojení. Toto znamená konec závodu a ukončení komunikace se serverem.

A.6 Popis vstupů a výstupů neuronové sítě

Klient-řidič dostává v průběhu závodu informace o poloze auta převedené na vstupy neuronové sítě:

- distance – vzdálenost auta od středové čáry.
  - 0 až 0,5 – auto je nalevo od čáry
  - 0,5 – auto je přesně na čáře
  - 0,5 až 1 – auto je napravo od čáry
- angle – úhel mezi čárou a směrem jízdy
  - 0 až 0,5 – auto je natočené nalevo
  - 0,5 – auto jede rovnoběžně s čárou
  - 0,5 až 1 – auto je natočené napravo
- speed - relativní rychlost auta
  - 0 až 0,5 – auto se pohybuje dozadu
  - 0,5 – auto stojí
  - 0,5 až 1 – auto se pohybuje dopředu
- distance2 – vzdálenost od čáry za určitou časovou jednotku, pokud bude auto pokračovat v jízdě stejným směrem

  - 0 až 0,5 – auto je nalevo od čáry
  - 0,5 – auto je přesně na čáře
  - 0,5 až 1 – auto je napravo od čáry
- friction – Součinitel tření mezi pneumatikou a vozovkou.
  - 0,2 – led
  - 0,5 – tráva
  - 1,0 – asfalt/beton
- skid – určuje zda je auto ve smyku
  - 0,0 – všechna kola jsou ve smyku
  - 0,5 – jedna náprava je ve smyku
  - 1,0 – všechna kola mají trakci
- checkpoint – směr k dalšímu checkpointu
  - 0.0 až 0.5 – checkpoint je nalevo od středu auta (-180° až 0°)
  - 0.5 – checkpoint je přímo před autem
  - 0.5 až 1.0 – checkpoint je nalevo od středu auta (0° až 180°)
- sensor1 – první senzor zjišťující překážky před autem
  - 0.0 – Nalezena překážka se středem nalevo od středové čáry.
  - 0.5 – Sensor nezjistil překážku.
  - 1.0 – Nalezena překážka se středem napravo od středové čáry.
- sensor2 – druhý senzor (širší) zjišťující překážky před autem
  - 0.0 – Nalezena překážka se středem nalevo od středové čáry.
  - 0.5 – Sensor nezjistil překážku.
  - 1.0 – Nalezena překážka se středem napravo od středové čáry.

Po odeslání informací o poloze auta očekává server odpověď od ridiče:

- wheel – natočení volantu
  - 0 až 0,5 – doleva
  - 0,5 – rovně
  - 0,5 až 1 – doprava
- acc – zrychlení
  - 0 – maximální brždění/zpátečka
  - 0,5 – pouze setrvačnost
  - 1 – maximální zrychlení

B Obsah CD

Přiložené CD obsahuje tyto adresáře a soubory:

- program – adresář se spustitelným programem Neural Racer
  - neurace.jar – spouštěč programu
  - client – datové soubory klienta
  - server – datové soubory serveru
  - databaze – konfigurační soubor databáze
- projekt – projekt pro IDE NetBeans včetně zdrojových souborů
- text – text práce a manuál k aplikaci
- javadoc – programátorská dokumentace
- basic_client – jednoduchý klient, který řídí auto.
  - BasicClient.jar – spustitelný soubor
  - driver.java – zdrojový kód klienta
- neural_client – klient, který umožňuje načítat neuronové sítě řídící auto.
  - NeuralNet.jar – spustitelný soubor
  - trenink – trénovací množiny (načíst a naučit)
  - naucene_site – již naučené sítě (načíst)